Serial No. 09/977,161
Docket No. TUC920010021US1
Firm No. 0018.0091

Amendments to the Specification:

Please replace the following paragraphs:

[0022] FIG. 4 illustrates logic implemented in the encoder 4 to allow a user to create an MRU key 10 that can be used as an encryption key to ~~decrypt the output data 12~~ encrypt the input data 8. Control begins at block 100 with the encoder 4 receiving a request from a user to encode data. In response, the encoder 4 generates (at block 102) an MRU key 10 with 256 one byte entries including every possible binary permutation of an eight bit byte. The encoder 4 then rearranges (at block 104) the entries in the MRU key 10. The encoder 4 may user a random number or other random data to reorder the entries. The result is a randomized MRU key 10. Alternatively, the encoder 4 may receive a secret password from the user and use the value of the password to generate an ordering of the entries in the MRU key 10. The encoder 4 then returns (at block 106) the rearranged or generated MRU key 10 to the user. At block 108, the encoder 4 then encodes the data using the logic of FIG. 3 and the generated MRU key 10 and outputs (at block 110) the encoded data. The user may store the MRU key 10 for use to decode the data according to the logic of FIG. 5, or alternately just some password used to generate the key may be stored so that it can be regenerated for decryption.

[0026] FIG. 6 illustrates a further implementation where additional operations are performed to further encrypt the input data to increase data security and further confound third parties improperly attempting to decode the data. The implementation of FIG. 6 includes all the components of FIG. 1 and additionally includes four random number generators (RNG) 230, 232, 234, and 235, which are used by the encoder 204, decoder 206, and bit packer 236. The random number generators 230, 232, 234, and 235 implement pseudo random number generator algorithms known in the art. Further, a scrambler 238 performs scrambling operations in response to the value of an input pseudo random number from one of the random number generators 230, 232, 234, or 235. A bit packer 236 gathers bits of the encoded data and packs the bits into N bit

packages, such as 32 bit packages, to generate into the output data 212 in a manner known in the art.

[0027] FIG. 7 illustrates logic implemented in the encoder 6, that uses many of the steps of FIG. 3 and the random number generators 230, 232, 234, and 235 to further encrypt the input data 208 during the compression operations in order to increase the security of the encoded output data 12. Control begins at block 300 where the encoder 204 begins the process of encoding the input data 208. The encoder 204 receives (at block 302) the scrambled MRU key 210, initializes the MRU list 220 with the content of the received MRU key 210, and seeds all four random number generators 230, 232, 234, and 235 with one or more bits from the MRU key 210. Seeding the pseudo random number generators 230, 232, 234, and 235 with the MRU key 210, which is itself secure, further obstructs third parties from decoding the output data 212. At block [[206]] 306, the encoder 4 then scans a string from the input data stream 208 and generates (at block 308) a copy pointer or MRU reference for literals included in the string according to steps 52 through 66 68 in FIG. 3.

[0030] FIG. 8 illustrates the operations performed by the bit packer 236 upon receiving (at block 350) the copy pointers or MRU references from the encoder logic 204. The bit packer 236 collects (at block 352) compression codewords until N (e.g. 32, as will be assumed below) contiguous bits of the compressed data stream exist which can be output at one time. Certain special conditions, such as the end of a record, or operations may cause all bits in the bit packer to be output even when there are not 32 valid bits -- in this case the as yet undefined bits can be set to some default value, such as binary '0's, or they can be set to some part of a random number. In either case, the output of the bit packer is 32 bit segments of the compressed data stream. Each 32 bit output of the bit packer is then encrypted in two ways.